Ideas for Finer-grained Control over your Heat Budget

ELC NA 2020: **Corona Lockdown edition**

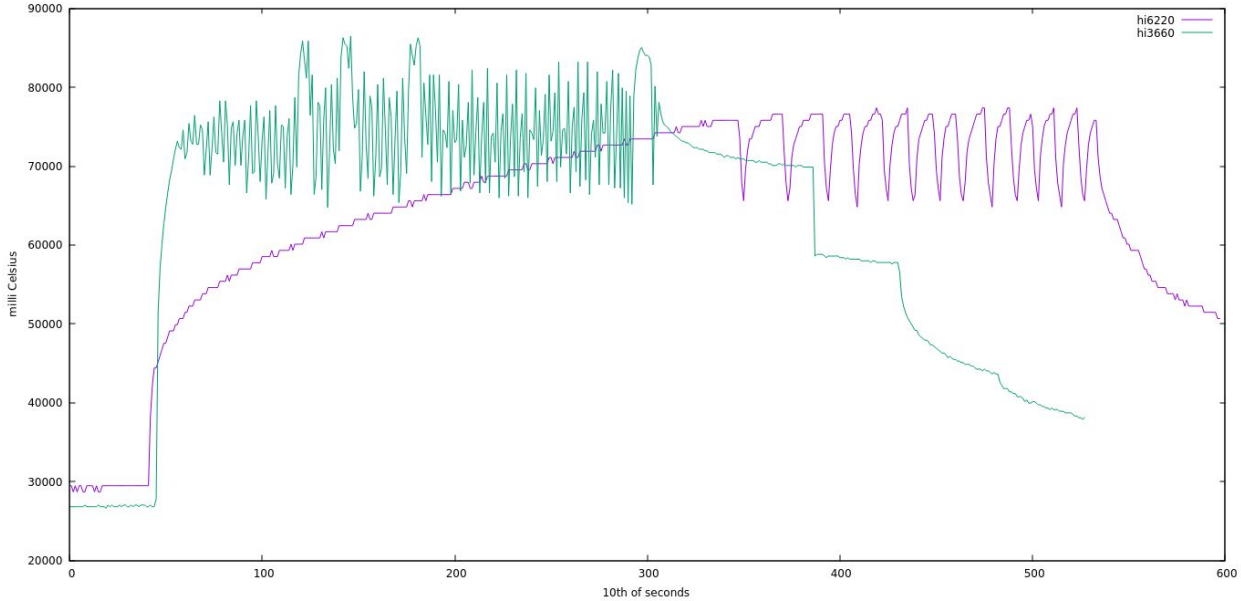
Amit Kucheria, Daniel Lezcano
Linaro

Outline

- Terminology
- Goals
- Linux Thermal Framework
- Real-world usage
- Limitations
- Our Proposal
- Future work

Linux thermal framework

Tries to cap the temperature given a specific policy in order to prevent hotspots



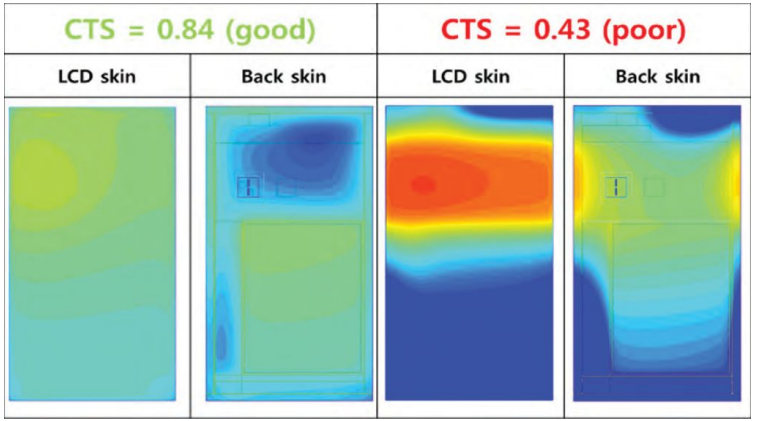
Linaro

Term: Junction temperature limit

- Temperature threshold (T_j) beyond which the silicon may get damaged
- Thermal runaway phenomena makes heating speed quadratic
- Depends on manufacturing process e.g. 120°C on some Qualcomm SoCs
- Firmware *may* trigger emergency shutdown

Term: Skin Temperature limit

- Temperature threshold (T_{skin}) beyond which device is too hot to touch (~45°C)
- User non-contact surfaces can allow higher T_{skin} e.g. base of ~~laptop~~ notebook
- Standards: IEC 60950 or IEC 62368 at 25°C ambient
- Form factor design distributes the heat
  - Larger form factor = easier distribution



Source: A Figure of Merit for Smart Phone Thermal Management

Term: Thermal Design Power

- TDP usually used to allocate thermal budget to sustain a performance

$$TDP\ (watts) = (T_{case} - T_{ambient}) / \Theta_{case}$$

- Θ_{case} : Thermal resistivity for the dissipation device (°C / W)
- Higher the thermal resistivity, lower the dissipated power
- Higher the ambient temperature, lower the dissipation budget

Goals for TDP management

- Fixed TDP devices
  - Passively cooled
  - Operate the device under the TDP budget (W) for as long as possible to stay under T_{skin} temperature threshold → Sustained performance
  - Dynamically balance the total dissipation budget across all heat generators
- Flexible TDP devices
  - Actively cooled
  - All of the above goals
  - Enhance TDP limits using external help for dissipation
    - Internal: fans, liquid cooling
    - External: air-conditioning

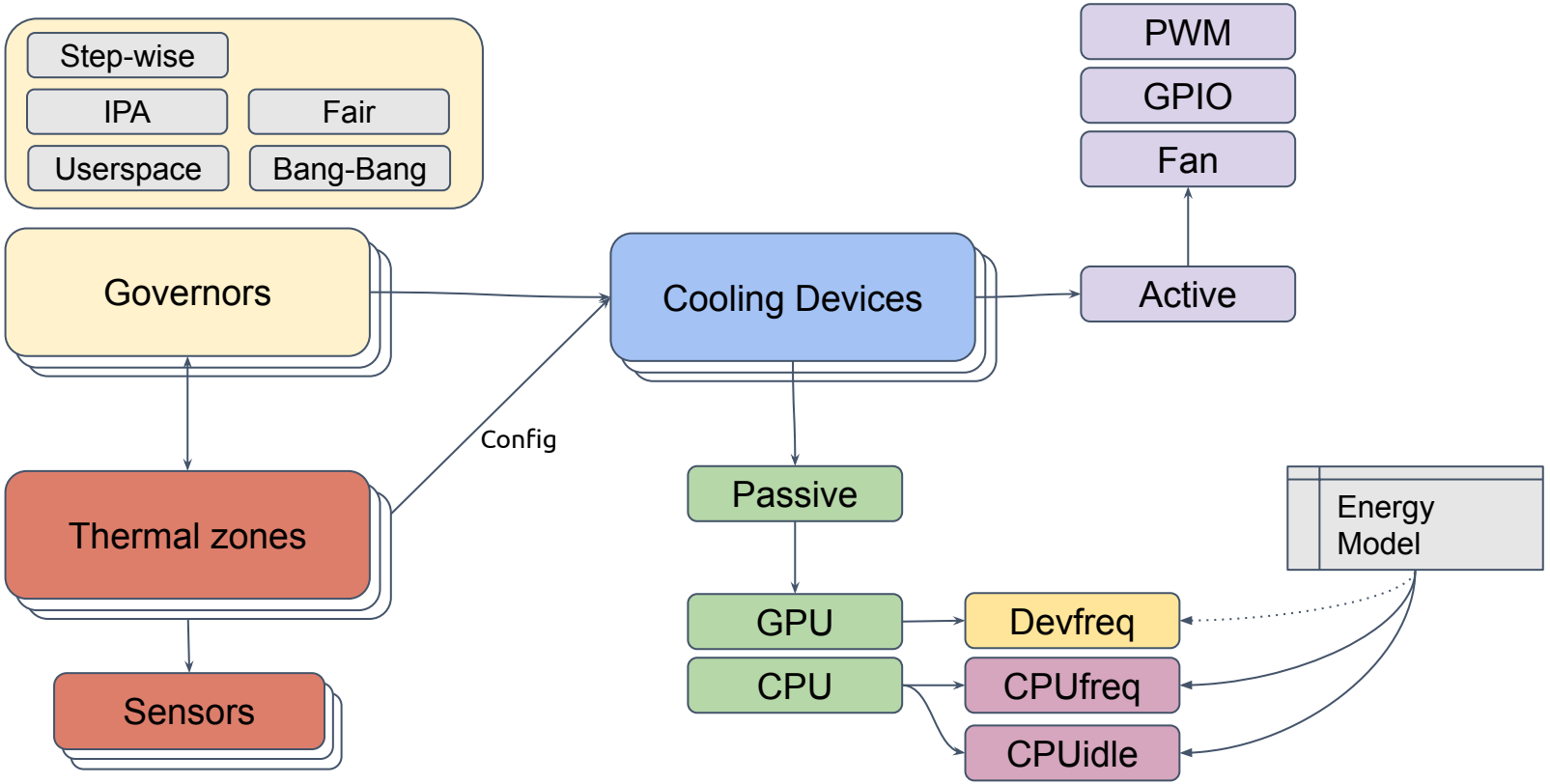
Major sources of heat

- CPU
- GPU
- 4G / 5G Modem, WiFi
- DSP/Accelerators
- NPU
- Camera
- Memory

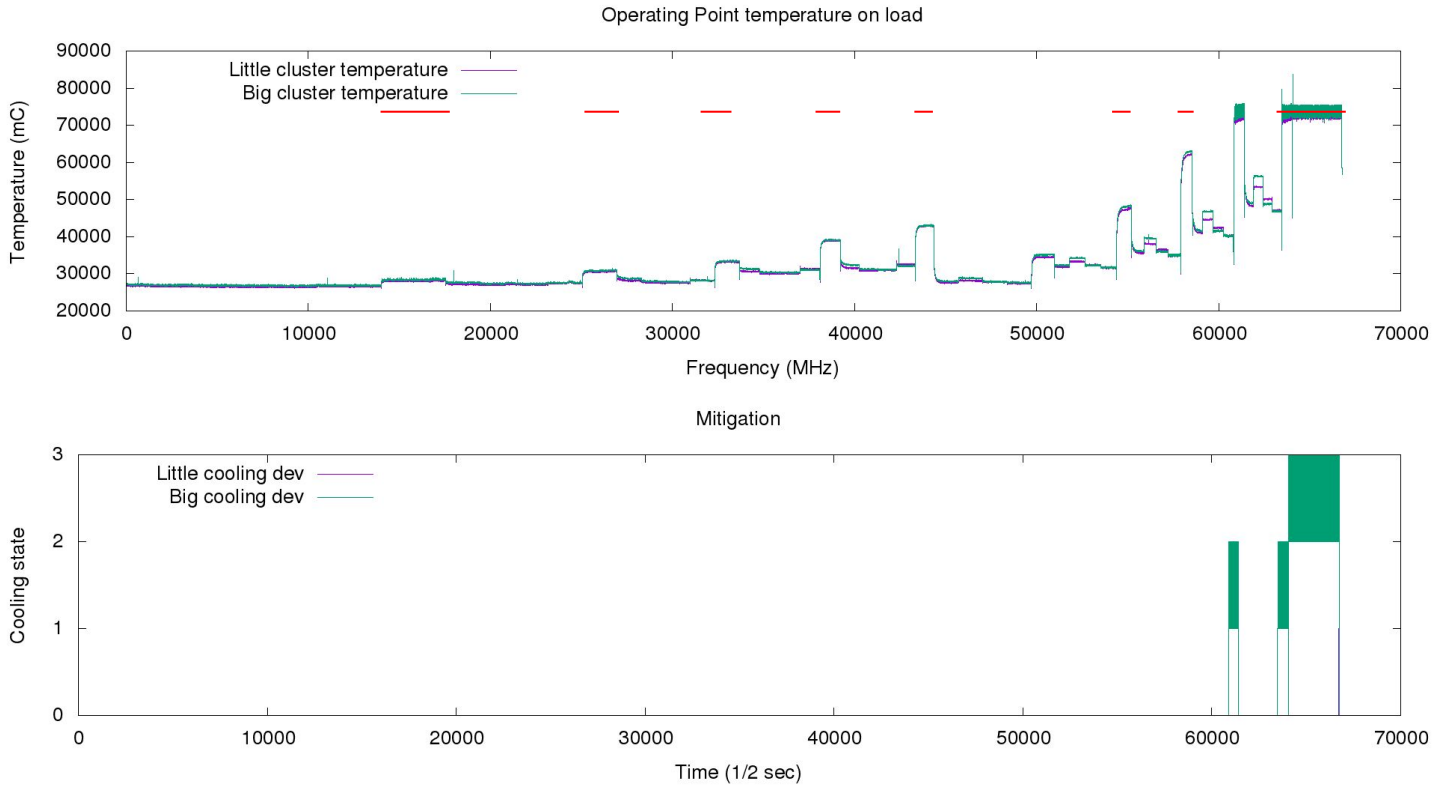
Challenging use cases

- Virtual/Augmented reality
  - Headsets with 4K resolution @ 90-120 Hz refresh-rates coming
  - Limited thermal budget, if crossed it'll cause throttling which leads to frame drops causing dizziness
  - Allow GPU and memory a bigger share of the thermal budget
- Gaming: GPU gets more budget compared to CPU
- Phone: Modem gets priority over others
- Streaming Movie: GPU, Video decoder, Wifi/4G, CPU all competing
- Web browsing: CPU, GPU, Wifi/4G all competing

Linux thermal framework



Correlation: cooling state and temperature



Real-world usage

- CPUs frequency-throttled based on sensor input for T_j and T_{skin} management
- Userspace thermal daemons rely on thermal framework interfaces
  - Read temperature of each sensor from sysfs
  - Set *opaque* cooling device state (1, 2, 3, ..)
- The daemons may use knowledge of use case to throttle specific devices
  - Throttle CPU so more TDP available to GPU and other actors
  - Send hint to GPU to drop resolution
  - Send hints to non-Linux devices to reduce performance if possible (e.g. modem)

Limitations

- Entire cluster can get throttled
  - Might unnecessarily kill performance
  - Using idle-injection for individual CPUs and frequency throttling for the cluster?
- Userspace thermal daemons set *opaque* cooling device state (1, 2, 3, ..)
- Userspace thermal daemons and in-kernel governor compete for decisions
  - In some cases, userspace simply overrides kernel governors
- In-kernel governors don't talk to each other
- *Observation*: T_{skin} management needs TDP balancing (watts!), not direct temperature management

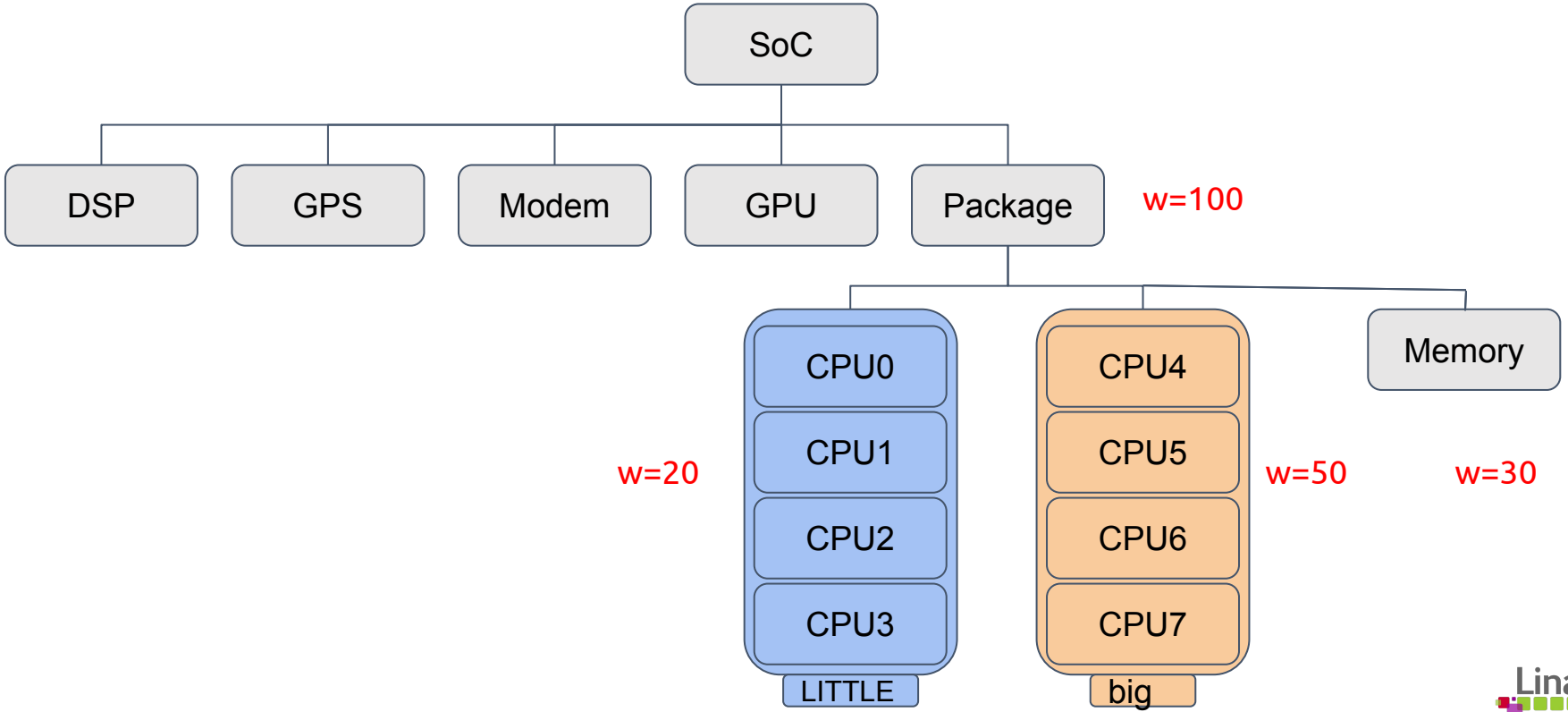
Our Proposal

- Separation of thermal management and power capping
- Handle TDP balancing in Powercap using the energy model instead of using opaque cooling device states to try to hit the right performance state for a device
  - Use case-based throttling is performance capping, belongs outside thermal framework
- Kernel-based T_j management → Thermal Framework
- User space hints for T_{skin} management → Powercap Framework

Power Capping framework

- The Linux kernel provides the power capping framework
  - Generic sysfs interface : /sys/devices/virtual/powercap/
- The power capping controller implements a solution as backend driver
  - One controller Intel RAPL for CPU/Memory energy power/limitation
- Hierarchical constraints already supported
- Userspace tools and library available

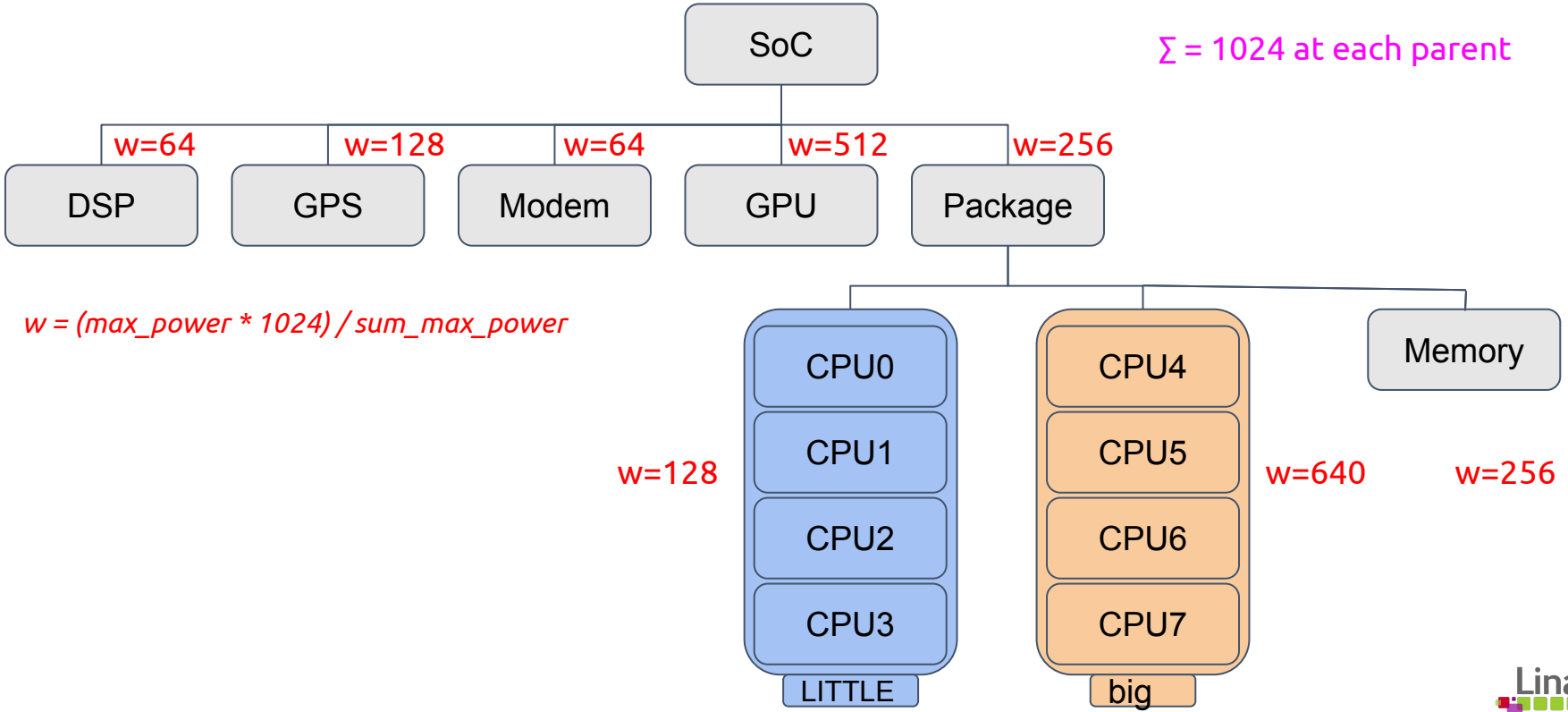
Power Capping using Energy Model



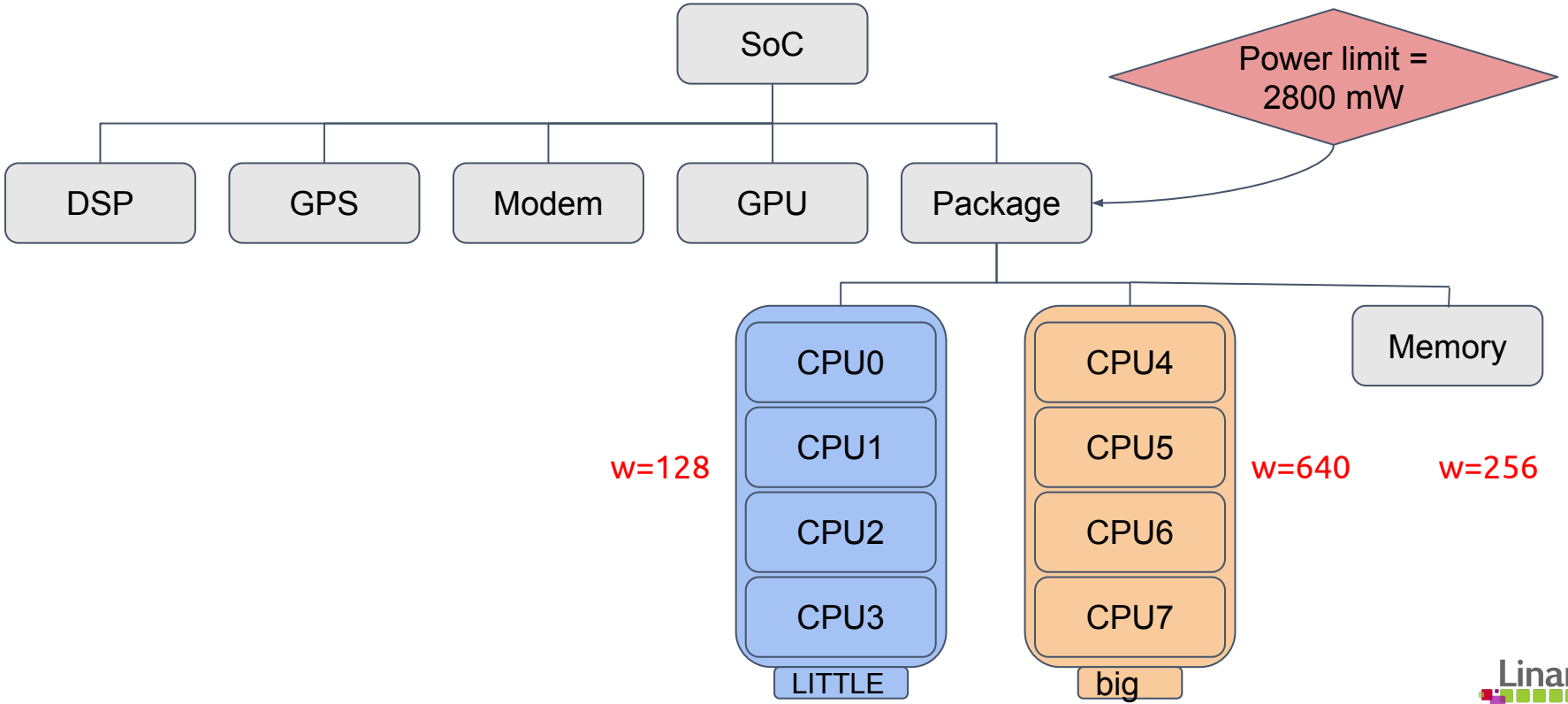
Power Capping using Energy Model

- The Energy model is supported and provides enough information
  - Performance state ↔ power
- Model the constraints with the powercap hierarchy
- The parent node’s max power consumption and limitations is the sum of the children’s max power and limitations
- Power is redistributed across sibling nodes if unused by a node
- Each leaf node has a max power consumption
- Userspace thermal daemons can guarantee the skin temperature by setting the constraints

Weight for fair power limit redistribution



Weight for fair power limit redistribution



Weight for fair power limit redistribution

Devices	***Weight***	***Static Limit (mW)***
LITTLE	128	350
big	640	1750
Memory	256	700
Total	1024	2800

Unused power redistribution among siblings

Devices	***Weight***	***Static Limit (mW)***	***Dynamic Limit (mW)***	***Used (mW)***	***Free (mW)***
LITTLE	128	350	350	350	0
big	640	1750	1750	300	1450
Memory	256	700	700	700	0
Total	1024	2800	2800	1350	1450

Unused power redistribution among siblings

Devices	***Weight***	***Static Limit (mW)***	***Dynamic Limit (mW)***	***Used (mW)***	***Free (mW)***
LITTLE	128	350	350+483	350	0
big	640	1750	1750-1450	300	1450
Memory	256	700	700+967	700	0
Total	1024	2800	2800	1350	1450

Linaro

Unused power redistribution among siblings

Devices	***Weight***	***Static Limit (mW)***	***Dynamic Limit (mW)***	***Used (mW)***	***Free (mW)***
LITTLE	128	350	833	350	483
big	640	1750	300	300	0
Memory	256	700	1667	700	967
Total	1024	2800	2800	1350	1450

Unused power redistribution among siblings

Devices	***Weight***	***Static Limit (mW)***	***Dynamic Limit (mW)***	***Used (mW)***	***Free (mW)***
LITTLE	128	350	833	↑ 350	483
big	640	1750	300	300	0
Memory	256	700	1667	700	967
Total	1024	2800	2800	1350	1450

Unused power redistribution among siblings

Devices	***Weight***	***Static Limit (mW)***	***Dynamic Limit (mW)***	***Used (mW)***	***Free (mW)***
LITTLE	128	350	833	766	67
big	640	1750	300	300	0
Memory	256	700	1667	700	967
Total	1024	2800	2800	1766	1034

On demand power rebalancing

Devices	***Weight***	***Static Limit (mW)***	***Dynamic Limit (mW)***	***Used (mW)***	***Free (mW)***
LITTLE	128	350	833	766	67
big	640	1750	300	↑ 300	0
Memory	256	700	1667	700	967
Total	1024	2800	2800	1766	1034

Linaro

On demand power rebalancing

Devices	***Weight***	***Static Limit (mW)***	***Dynamic Limit (mW)***	***Used (mW)***	***Free (mW)***
LITTLE	128	350	833	766	67
big	640	1750	300	1450	0
Memory	256	700	1667	700	967
Total	1024	2800	2800	**2916**	1034

On demand power rebalancing

Devices	***Weight***	***Static Limit (mW)***	***Dynamic Limit (mW)***	***Used (mW)***	***Free (mW)***
LITTLE	128	350	350	766	0
big	640	1750	1750	1450	300
Memory	256	700	700	700	0
Total	1024	2800	2800	**2916**	300

On demand power rebalancing

Devices	***Weight***	***Static Limit (mW)***	***Dynamic Limit (mW)***	***Used (mW)***	***Free (mW)***
LITTLE	128	350	350+100	766	0
big	640	1750	1750-300	1450	300
Memory	256	700	700+200	700	0
Total	1024	2800	2800	**2916**	300

On demand power rebalancing

Devices	***Weight***	***Static Limit (mW)***	***Dynamic Limit (mW)***	***Used (mW)***	***Free (mW)***
LITTLE	128	350	450	766	0
big	640	1750	1450	1450	0
Memory	256	700	900	700	200
Total	1024	2800	2800	**2916**	200

On demand power rebalancing

Devices	***Weight***	***Static Limit (mW)***	***Dynamic Limit (mW)***	***Used (mW)***	***Free (mW)***
LITTLE	128	350	450 + 200	766	0
big	640	1750	1450	1450	0
Memory	256	700	900 - 200	700	200
Total	1024	2800	2800	**2916**	200

On demand power rebalancing

Devices	***Weight***	***Static Limit (mW)***	***Dynamic Limit (mW)***	***Used (mW)***	***Free (mW)***
LITTLE	128	350	650	766	0
big	640	1750	1450	1450	0
Memory	256	700	700	700	0
Total	1024	2800	2800	**2916**	0

On demand power rebalancing

Devices	***Weight***	***Static Limit (mW)***	***Dynamic Limit (mW)***	***Used (mW)***	***Free (mW)***
LITTLE	128	350	650	→ 650	0
big	640	1750	1450	1450	0
Memory	256	700	700	700	0
Total	1024	2800	2800	2800	0

Power Capping using Energy Model

- Multiple constraints in the hierarchy
  - Fine-grained control on automatic power redistribution, overriding node weight
- Allow power limit violation during burst
  - Absorb peak load (eg. web page loading)
- Power limit timeout
  - Constraint automatically removed after a duration. That will allow simplification of the userspace code.

Status: Energy Model

Ongoing	• Lukasz Luba's work to generalize the Energy Model in the device structure • Devfreq using the energy model	https://lkml.org/lkml/2020/5/27/406
Future	• Power QoS • Add set/get power ops • Improve GPU drivers to use the energy model	Interdependent devices belonging to the same voltage / freq domain Same interface to aggregate the requests Provide the same interface Unify load usage of the GPU for dynamic power ratio

Status: Power Capping using Energy Model

Ongoing	• Prototype with automatic power rebalancing • Power vs performance measurements • Solution comparisons	
Future	• More devices supporting the energy model • DT description	Modem, Accelerators unknown to Linux
Challenges	• Inclusion of non-Linux devices on SoC	

Conclusion

- The thermal framework: main goal is to protect the silicon
- Complex SoCs need to manage the heating devices as a whole for skin temperature
- The powercap framework with the Energy Model
  - Model the constraints
  - Delegate the logic to userspace
  - Watt-centric
  - Extensible solution

Thank you

and stay safe!

amit.kucheria@linaro.org
daniel.lezcano@linaro.org